Analysis of derivative-free trust-region methods based on probabilistic models

Carlos A. Sing-Long

CME334: Advanced Methods in Numerical Optimization

October 2013

The following presentation is based on

[1] A. S. Bandeira, K. Scheinberg, L. N. Vicente, "Convergence of trust-region methods based on probabilistic models", arXiv:1304.2808, 2013.

[2] A. S. Bandeira, K. Scheinberg, L. N. Vicente, "Computation of sparse low-degree interpolating polynomials and their application to derivative-free optimization", arXiv:1306.5729, 2013.

Derivative-free methods

Model-based trust-region methods

Models

Trust-region methods based on fully-linear models

Trust-region methods based on fully-quadratic models

Building the model

Concluding remarks

Derivative-free methods

Model-based trust-region methods

Models

Trust-region methods based on fully-linear models

Trust-region methods based on fully-quadratic models

Building the model

Concluding remarks

Motivation

Here we will consider numerical methods for unconstrained optimization problems, i.e., how to build sequences $\{x_k\}$ converging to the optimal point of

$$\min_{x \in \mathbb{R}^n} \quad f(x)$$

where f is a sufficiently regular function bounded below.

Motivation

In many cases we cannot compute the gradient of f even though it exists. For instance

- The objective function could be a black-box.
- A closed-form expression is available, but it is computationally intractable or unreasonably expensive to compute.
- Approximation of the gradient using finite differences can be inaccurate or computationally expensive.

Motivation

In some problems of interest the objective function is the result of a computationally intensive simulation.

- In circuit design one wants to minimize the interference between different components, power dissipation, and leakage current. These quantities are computed through simulations in PSpice, which can be computationally expensive for large circuits operating for long periods of time.
- In the simulation of mechanical systems with potential contact by friction. Although the problem may have few variables, the function evaluations depend on the integration of a modified ODE system with potential nondifferentiabilities.

Examples taken from http://www.mat.uc.pt/~lnv/dfo.html

Motivation

In these cases it makes sense to use an optimization method that does not rely on gradient information.

Derivative-free methods

A **derivative-free** method builds a local model of the objective function based on sampled function values, or it directly exploits a sample set of function values without building an explicit model.

With this information, the method generates the next iterate or determines the optimality of the current iterate.

Derivative-free methods

In practice, these methods should be used only when very little information about f can be obtained, or the information we have about it is corrupted.

Derivative-free methods

In practice, these methods should be used only when very little information about f can be obtained, or the information we have about it is corrupted.

"If you can obtain clean derivatives (even if it requires considerable effort) and the functions defining your problem are smooth and free of noise you should not use derivative-free methods"

A. R. Cohn, K. Scheinberg, L. N. Vicente, *Introduction to Derivative-Free Optimization*

Derivative-free methods

Model-based trust-region methods

Models

Trust-region methods based on fully-linear models

Trust-region methods based on fully-quadratic models

Building the model

Concluding remarks

Trust-region methods

A trust-region method consists of building a model for f over a neighborhood of the current iterate. This model is minimized over the neighborhood, and the minimizer of this surrogate problem is chosen as the next iterate.

Trust-region methods

Typically, this means at x_0 we pick a radius $\delta > 0$ and use a **model** m for f over the ball $B(x_0, \delta)$. The model m is selected from a class $\mathcal{M}$ of possible functions to approximate f over the trust region. We solve

$$\min_{x \in B(x_0, \delta)} \quad m(x)$$

The minimizer of this problem is chosen as the next iterate if certain **acceptance criteria** are met.

Trust-region methods

- Which properties ensure the class $\mathcal{M}$ has accurate models? Are these models "easy" to optimize?
- How should we choose the model?
- Which criteria should we use to accept the minimizer as the next iterate? What do we do when it is rejected?
- Does the resulting method converge?

Derivative-free methods

Model-based trust-region methods

Models

Trust-region methods based on fully-linear models

Trust-region methods based on fully-quadratic models

Building the model

Concluding remarks

Linear and quadratic models

The natural criteria for a good or well-behaved model is that it should approximate f at least as well as a Taylor expansion of low order.

Fully linear models

For a trust-region $B(x_0, \delta)$ we say that a model m is **fully linear** if

$$\begin{aligned} \|\nabla f(x) - \nabla m(x)\| &\leq \kappa_g \delta \quad \forall\, x \in B(x_0, \delta) \\ |f(x) - m(x)| &\leq \kappa_f \delta^2 \quad \forall\, x \in B(x_0, \delta) \end{aligned}$$

If $\mathcal{M}$ is fully linear, then κ_g, κ_f are assumed to be independent of δ and x_0.

Fully linear models

For a trust-region $B(x_0, \delta)$ we say that a model m is **fully linear** if

$$\begin{aligned} \|\nabla f(x) - \nabla m(x)\| &\leq \kappa_g \delta \quad \forall\, x \in B(x_0, \delta) \\ |f(x) - m(x)| &\leq \kappa_f \delta^2 \quad \forall\, x \in B(x_0, \delta) \end{aligned}$$

If $\mathcal{M}$ is fully linear, then κ_g, κ_f are assumed to be independent of δ and x_0.

This is a generalization of a **first-order** Taylor expansion

$$m(x) = f(x_0) + \nabla f(x_0)^T (x - x_0)$$

Fully quadratic models

For a trust-region $B(x_0, \delta)$ we say that a model m is **fully quadratic** if

$$\begin{aligned} \|\nabla^2 f(x) - \nabla^2 m(x)\| &\leq \kappa_h \delta \quad \forall\, x \in B(x_0, \delta) \\ \|\nabla f(x) - \nabla m(x)\| &\leq \kappa_g \delta^2 \quad \forall\, x \in B(x_0, \delta) \\ |f(x) - m(x)| &\leq \kappa_f \delta^3 \quad \forall\, x \in B(x_0, \delta) \end{aligned}$$

If $\mathcal{M}$ is fully quadratic, κ_g, κ_f and κ_h are assumed to be independent of δ and x_0.

Fully quadratic models

For a trust-region $B(x_0, \delta)$ we say that a model m is **fully quadratic** if

$$\begin{aligned} \|\nabla^2 f(x) - \nabla^2 m(x)\| &\leq \kappa_h \delta \quad \forall\, x \in B(x_0, \delta) \\ \|\nabla f(x) - \nabla m(x)\| &\leq \kappa_g \delta^2 \quad \forall\, x \in B(x_0, \delta) \\ |f(x) - m(x)| &\leq \kappa_f \delta^3 \quad \forall\, x \in B(x_0, \delta) \end{aligned}$$

If $\mathcal{M}$ is fully quadratic, κ_g, κ_f and κ_h are assumed to be independent of δ and x_0.

This is a generalization of a **second-order** Taylor expansion

$$m(x) = f(x_0) + \nabla f(x_0)^T (x - x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0)$$

Assumptions

We will assume the models in $\mathcal{M}$ are at least twice-differentiable with Lipschitz second-derivatives, and that

$$\sup_{\substack{x \in \mathbb{R}^n \\ m \in \mathcal{M}}} \|\nabla^2 m(x)\| < \infty$$

Probabilistic models

With a criteria for well-behaved class of approximations, we can decide how to choose a model from $\mathcal{M}$. In many cases our choice of model m for a given trust region will not be deterministic.

Probabilistic models

With a criteria for well-behaved class of approximations, we can decide how to choose a model from $\mathcal{M}$. In many cases our choice of model m for a given trust region will not be deterministic.

- For instance, we could construct linear/quadratic models by interpolating m on a random set of samples Y belonging to the trust-region $B(x_0, \delta)$.

Probabilistic models

With a criteria for well-behaved class of approximations, we can decide how to choose a model from $\mathcal{M}$. In many cases our choice of model m for a given trust region will not be deterministic.

- For instance, we could construct linear/quadratic models by interpolating m on a random set of samples Y belonging to the trust-region $B(x_0, \delta)$.
- The values of the objective could be corrupted by noise. For instance, due to being a Monte Carlo simulation, or due to the presence of numerical inaccuracies in a simulation.

Probabilistic models

In these cases the optimization algorithm will produce a sequence of random iterates $\{X_k\}$, random radii $\{\Delta_j\}$, and random models $\{M_k\}$.

Therefore it makes sense to introduce a notion of a choice of well-behaved models for this setting.

Probabilistic models

We say that a sequence of $\{M_k\}$ of random models is p-probabilistically (κ_f, κ_g)-fully linear/$(\kappa_f, \kappa_g, \kappa_h)$-fully quadratic for a corresponding sequence $\{B(X_k, \Delta_k)\}$ if the events

$$S_k = \left\{ \begin{array}{l} M_k \text{ is a } (\kappa_f, \kappa_g)\text{-fully linear}/(\kappa_f, \kappa_g, \kappa_h)\text{-fully quadratic} \\ \text{model of } f \text{ on } B(X_k, \Delta_k) \end{array} \right\},$$

satisfy

$$\mathbf{P}\{S_k \mid M_{k-1}, \ldots, M_0\} \geq p.$$

In other words, the proportion of iterations yielding good models is **bounded below**.

Derivative-free methods

Model-based trust-region methods

Models

Trust-region methods based on fully-linear models

Trust-region methods based on fully-quadratic models

Building the model

Concluding remarks

Criteria for sufficient decrease for fully-linear models

If $\mathcal{M}$ is fully-linear, we will assume that for any x_0 and δ we can always find s with $x_0 + s \in B(x_0, \delta)$ such that

$$m(x_0) - m(x_0 + s) \geq \frac{\kappa_d}{2} \|\nabla m(x_0)\| \min \left\{ \frac{\|\nabla m(x_0)\|}{\|\nabla^2 m(x_0)\|}, \delta \right\}$$

for some $0 < \kappa_d \leq 1$ independent of m. If s satisfies the above, then we say it has achieved a fraction of **Cauchy decrease**.

The algorithm for fully-linear models

Let x_0,$\eta_1, \eta_2, \gamma > 0$ and $\delta_{\max} > 0$ with $\gamma > 1$ and $\eta_1 < 1$ be inputs. At the k-th iteration, choose a model m_k for f on $B(x_k, \delta_k)$ and minimize m_k over $B(x_k, \delta_k)$ so that the step s_k satisfies a fraction of Cauchy decrease. Let

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m(x_k) - m(x_k + s_k)}$$

and set

$$(x_{k+1}, \delta_{k+1}) = \begin{cases} (x_k + s_k, \min\{\gamma\delta_k, \delta_{\max}\}) & \text{if } \rho_k \geq \eta_1 \\ & \text{and } \|\nabla m_k(x_k)\| \geq \eta_2 \delta_k, \\ (x_k, \gamma^{-1}\delta_k) & \text{otherwise.} \end{cases}$$

Analysis of convergence

Lemma (Lemma 3.1 [1])

For every realization of the algorithm we have

$$\lim_{k \to \infty} \delta_k = 0.$$

Analysis of convergence

Lemma (Lemma 3.1 [1])

For every realization of the algorithm we have

$$\lim_{k \to \infty} \delta_k = 0.$$

The proof relies uniquely on the existence of a step of Cauchy decrease for every realization of the process.

Analysis of convergence

Lemma (Lemma 3.2 [1])

If $\{m_k\}$ *is* (κ_f, κ_g)*-fully linear on* $B(x_k, \delta_k)$ *and*

$$\delta_k \leq \|\nabla m(x_k)\| \min \left\{ \frac{1}{\sup_{\substack{x \in \mathbb{R}^n \\ m \in \mathcal{M}}} \|\nabla^2 m(x)\|}, \frac{\kappa_d(1-\eta_1)}{4\kappa_f} \right\},$$

then at the k-th iteration $\rho_k \geq \eta_1$.

Analysis of convergence

Lemma (Lemma 3.2 [1])

If $\{m_k\}$ *is* (κ_f, κ_g)*-fully linear on* $B(x_k, \delta_k)$ *and*

$$\delta_k \leq \|\nabla m(x_k)\| \min \left\{ \frac{1}{\sup_{\substack{x \in \mathbb{R}^n \\ m \in \mathcal{M}}} \|\nabla^2 m(x)\|}, \frac{\kappa_d(1-\eta_1)}{4\kappa_f} \right\},$$

then at the k-th iteration $\rho_k \geq \eta_1$.

Roughly speaking, the algorithm eventually accepts a new iterate unless it has reached a critical point.

Analysis of convergence

Lemma (Theorem 4.3 (1))

Suppose $\{M_k\}$ *is (1/2)-probabilistically* (κ_f, κ_g)*-fully linear on* $B(X_k, \Delta_k)$ *where* $\{(X_k, \Delta_k)\}$ *are the iterates of the trust-region algorithm. Then*

$$\lim_{k \to \infty} \|\nabla f(X_k)\| = 0$$

almost surely.

Analysis of convergence

This result improves the existing ones in that the convergence is **almost sure**. Previously only results in expectation were known.

However this result does not provide rates of convergence. It is intuitive that as the probability of obtaining a full-linear model at each iteration increases, the performance of the algorithm will be comparable to that of a gradient-based method.

Derivative-free methods

Model-based trust-region methods

Models

Trust-region methods based on fully-linear models

Trust-region methods based on fully-quadratic models

Building the model

Concluding remarks

Criteria for sufficient decrease for fully-quadratic models

In this case we need to modify the criteria, as we need to account for directions of negative curvature on the Hessian. We assume for fully-quadratic models that we can find s with $x_0 + s \in B(x_0, \delta)$ such that

$$m(x_0) - m(x_0 + s) \geq \frac{\kappa_d}{2} \max \left\{ \begin{array}{c} \|\nabla m(x_0)\| \min \left\{ \frac{\|\nabla m(x_0)\|}{\|\nabla^2 m(x_0)\|}, \delta \right\} \\ \max\{-\lambda_{\min}(\nabla^2 m(x_0)), 0\}\delta^2 \end{array} \right\}$$

for some $0 < \kappa_d \leq 1$ independent of m. If s satisfies the above, then we say it has achieved a fraction of **optimal decrease**.

Note such a step can be expensive to compute in practice.

Criteria for sufficient decrease for fully-quadratic models

In this case we need to modify the criteria, as we need to account for directions of negative curvature on the Hessian. We assume for fully-quadratic models that we can find s with $x_0 + s \in B(x_0, \delta)$ such that

$$m(x_0) - m(x_0 + s) \geq \frac{\kappa_d}{2} \max \left\{ \begin{array}{c} \|\nabla m(x_0)\| \min \left\{ \frac{\|\nabla m(x_0)\|}{\|\nabla^2 m(x_0)\|}, \delta \right\} \\ \max\{-\lambda_{\min}(\nabla^2 m(x_0)), 0\}\delta^2 \end{array} \right\}$$

for some $0 < \kappa_d \leq 1$ independent of m. If s satisfies the above, then we say it has achieved a fraction of **optimal decrease**.

Note such a step can be expensive to compute in practice.

The algorithm for fully-quadratic models

The criteria to accept or reject a new iterate also needs to incorporate information about directions of negative curvature. Define

$$\tau(x, f) = \max \left\{ \min \left\{ \|\nabla f(x)\|, \frac{\|\nabla f(x)\|}{\|\nabla^2 f(x)\|} \right\}, -\lambda_{\min}(\nabla^2 f(x)) \right\},$$

and define $\tau(x, m)$ similarly.

The algorithm for fully-quadratic models

Let x_0,$\eta_1, \eta_2, \gamma > 0$ and $\delta_{\max} > 0$ with $\gamma > 1$ and $\eta_1 < 1$ be inputs. At the k-th iteration, choose a model m_k for f on $B(x_k, \delta_k)$ and minimize m_k over $B(x_k, \delta_k)$ so that the step s_k satisfies a fraction of optimal decrease. Let

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m(x_k) - m(x_k + s_k)}$$

and set

$$(x_{k+1}, \delta_{k+1}) = \begin{cases} (x_k + s_k, \min\{\gamma\delta_k, \delta_{\max}\}) & \text{if } \rho_k \geq \eta_1 \\ & \text{and } \tau(x_k, m_k) \geq \eta_2\delta_k, \\ (x_k, \gamma^{-1}\delta_k) & \text{otherwise.} \end{cases}$$

Analysis of convergence

Lemma (Lemma 5.4 [1])

For every realization of the algorithm we have

$$\lim_{k \to \infty} \delta_k = 0.$$

Analysis of convergence

Lemma (Lemma 3.2 [1])

If $\{m_k\}$ *is* $(\kappa_f, \kappa_g, \kappa_h)$*-fully quadratic on* $B(x_k, \delta_k)$ *and*

$$\delta_k \leq \tau(x_k, m_k) \min\left\{1, \sqrt{\frac{\kappa_d(1-\eta_1)}{4\kappa_f}\frac{1}{\tau(x_k, m_k)}}, \frac{\kappa_d(1-\eta_1)}{4\kappa_f}\right\},$$

then at the k-th iteration $\rho_k \geq \eta_1$.

Analysis of convergence

Lemma (Lemma 3.2 [1])

If $\{m_k\}$ *is* $(\kappa_f, \kappa_g, \kappa_h)$*-fully quadratic on* $B(x_k, \delta_k)$ *and*

$$\delta_k \leq \tau(x_k, m_k) \min\left\{1, \sqrt{\frac{\kappa_d(1-\eta_1)}{4\kappa_f}\frac{1}{\tau(x_k, m_k)}}, \frac{\kappa_d(1-\eta_1)}{4\kappa_f}\right\},$$

then at the k-th iteration $\rho_k \geq \eta_1$.

Roughly speaking, the algorithm eventually accepts a step unless it has reached a **second order** stationary point or local minima.

Analysis of convergence

Lemma (Theorem 5.1 [1])

Suppose $\{M_k\}$ *is (1/2)-probabilistically* $(\kappa_f, \kappa_g, \kappa_h)$*-fully quadratic on* $B(X_k, \Delta_k)$ *where* $\{(X_k, \Delta_k)\}$ *are the iterates of the trust-region algorithm. Then*

$$\lim \inf_{k \to \infty} \tau(X_k, f) = 0$$

almost surely.

Analysis of convergence

Unfortunately in this case the result is much weaker.

One of the issues is that even though we may be close to a minimum, there is no guarantee that fully-quadratic models will be selected.

Derivative-free methods

Model-based trust-region methods

Models

Trust-region methods based on fully-linear models

Trust-region methods based on fully-quadratic models

Building the model

Concluding remarks

Interpolation

One approach to build a model in the trust region is to choose a set of points Y on $B(x_0, \delta)$ at each iteration and then **interpolate** f on these points.

This can lead to problems

- We need **at least** $(n+1)(n+2)/2$ function evaluations to fit a quadratic model using polynomial interpolation.
- The linear system associated with the interpolation can be ill-conditioned if Y is chosen poorly.

Random undersampling

An alternative to mitigate these problems is to draw samples at random within the trust region. It is known that this approach improves the condition number of the polynomial interpolation matrix.

To address the $O(n^2)$ function evaluations required to fit a fully-quadratic model, we can **undersample** and then **regularize** the interpolation.

Regularized interpolation

Let $\{1, \varphi_i^{(1)}, \varphi_j^{(2)}\}$ be a basis for quadratic models, where $\varphi_i^{(1)}$ is linear, and $\varphi_j^{(2)}$ is quadratic. We want to find coefficients $\{\alpha^{(0)}, \alpha_i^{(1)}, \alpha_j^{(2)}\}$ such that

$$f(y) = \alpha^{(0)} + \sum_i \alpha_i^{(1)} \varphi_i^{(1)}(y) + \sum_j \alpha_j^{(2)} \varphi_j^{(2)}(y) \qquad y \in Y,$$

where Y has $n+1 < p < (n+1)(n+2)/2$ points.

The idea is to have enough information to interpolate a fully-linear model robustly, but not enough to interpolate a fully-quadratic model.

Regularized interpolation

We can write

$$f = \alpha^{(0)} e + M(\Phi^{(1)}, Y)\alpha^{(1)} + M(\Phi^{(2)}, Y)\alpha^{(2)},$$

where $f_i = f(y_i)$, $M(\Phi^{(1)}, Y)_{ij} = \varphi_j^{(1)}(y_i)$ and $M(\Phi^{(2)}, Y)_{ij} = \varphi_j^{(2)}(y_i)$.

The system is **underdetermined** whenever $p < (n+1)(n+2)/2$.

We solve

$$\min_{\alpha^{(2)}} \|\alpha^{(2)}\| \quad \text{subject to} \quad f = \alpha^{(0)} e + M(\Phi^{(1)}, Y)\alpha^{(1)} + M(\Phi^{(2)}, Y)\alpha^{(2)}$$

for a suitable norm $\|\cdot\|$.

ℓ_1-regularization

In some applications, the Hessian of the objective function f will be such that only a fraction of its entries will have large magnitude. It is intuitive that we might be able to interpolate a fully-quadratic model with fewer than $(n+1)(n+2)/2$ coefficients.

ℓ_1-regularization

Inspired by Compressed Sensing, a recently proposed approach [2] interpolates by solving

$$\min_{\alpha^{(2)}} \|\alpha^{(2)}\|_1 \quad \text{subject to} \quad f = \alpha^{(0)} e + M(\Phi^{(1)}, Y)\alpha^{(1)} + M(\Phi^{(2)}, Y)\alpha^{(2)}$$

The ℓ_1-norm above only promotes sparsity of the quadratic part of the model m.

How good is this approximation in general? How good is this approximation when the Hessian of f is **sparse**? Or when only a sparse fully-quadratic model to f exists?

ℓ_1-regularization

In order to answer these questions, suppose the points on Y are drawn from the uniform probability distribution over $B_\infty(x_0, \delta)$ with

$$\frac{\log p}{p} \geq 9c_1(h+n+1)(\log(h+n+1))^2 \log((n+1)(n+2)/2),$$

for some universal constant $c_1 > 0$ and $0 \leq h < n(n+1)/2$. Furthermore, let Ψ be the basis of second-order polynomials on $B_\infty(x_0, \delta)$ given by

$$\psi = \begin{cases} \frac{3\sqrt{5}}{2\delta^2} x_i^2 - \frac{\sqrt{5}}{2} \\ \frac{3}{\delta^2} x_i x_j \\ \frac{\sqrt{3}}{\delta} x_i \\ 1, \end{cases}$$

and let $\mathbf{M}(\Psi, Y)$ be the corresponding interpolation matrix.

ℓ_1-regularization

Then, for any $\boldsymbol{\alpha}_0$ with at most $n + 1 + h$ coefficients the solution $\alpha^\star$ to

$$\min_{\alpha} \ \|\alpha^{(2)}\|_1 \quad \text{subject to} \quad \|M(\Psi, Y)\alpha - (M(\Psi, Y)\alpha_0 + \varepsilon)\|_2 \leq \eta,$$

where $\|\varepsilon\| \leq \eta$ satisfies

$$\|\alpha^\star - \alpha_0\| \leq \frac{c_2}{\sqrt{p}}\eta.$$

with probability $1 - n^{-\gamma \log p}$ for some universal constants $c_2, \gamma > 0$ (Corollary 4.1 [2]).

ℓ_1-regularization

Therefore, this approach

ℓ_1-regularization

Therefore, this approach

- Recovers a sparse Hessian if the function evaluations are noiseless,

ℓ_1-regularization

Therefore, this approach

- Recovers a sparse Hessian if the function evaluations are noiseless,
- Recovers a good approximation to the true coefficients if there is a small model mismatch, or noise on the observations,

ℓ_1-regularization

Therefore, this approach

- Recovers a sparse Hessian if the function evaluations are noiseless,
- Recovers a good approximation to the true coefficients if there is a small model mismatch, or noise on the observations,
- Recovers a **sparse** fully quadratic model if one exists.

Derivative-free methods

Model-based trust-region methods

Models

Trust-region methods based on fully-linear models

Trust-region methods based on fully-quadratic models

Building the model

Concluding remarks

Concluding remarks

- The methods presented show that in the context of derivative-free optimization, it is not necessary to produce accurate models at each iteration to prove convergence, as long as the probability of producing an accurate model is bounded below.
- These methods may be extended to analyze the performance of traditional optimization algorithms. For instance, when line search procedures fail to produce a step that satisfies sufficient decrease conditions, or when the quadratic approximation to the function is not accurate for a few iterations.

Concluding remarks

- Using ℓ_1-minimization to construct local, quadratic models with a sparse Hessian allows us to build simple models that nevertheless capture the curvature of the objective function.
- However, these approaches increase the computational burden of each iteration, and this can substantially impact the performance of the overall algorithm.

Questions?